Optimizing Sensing: From Water to the Web

Andreas Krause **Carlos Guestrin**

May 2009
CMU-ML-09-107

Carnegie Mellon

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
MAY 2009		**00-00-2009 to 00-00-2009**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Optimizing Sensing: From Water to the Web	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Carnegie Mellon University,School of Computer Science,Pittsburgh,PA,15213**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT
see report

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**21**	

Optimizing Sensing: From Water to the Web

Andreas Krause[†] **Carlos Guestrin**[*]

May 2009
CMU-ML-09-107

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Where should we place sensors to quickly detect contamination in drinking water distribution networks? Which blogs should we read to learn about the biggest stories on the web? These problems share a fundamental challenge: How can we obtain the most useful information about the state of the world, at minimum cost?

Such sensing, or active learning, problems are typically NP-hard, and were commonly addressed using heuristics without theoretical guarantees about the solution quality. In this paper, we present algorithms which efficiently find provably near-optimal solutions to large, complex sensing problems. Our algorithms exploit submodularity, an intuitive notion of diminishing returns, common to many sensing problems: the more sensors we have already deployed, the less we learn by placing another sensor. In addition to identifying the most informative sensing locations, our algorithms can handle more challenging settings, where sensors need to be able to reliably communicate over lossy links, where mobile robots are used for collecting data or where solutions need to be robust against adversaries and sensor failures.

We also present results applying our algorithms to several real-world sensing tasks, including environmental monitoring using robotic sensors, activity recognition using a built sensing chair, a sensor placement challenge, and deciding which blogs to read on the web.

[†] Division of Engineering and Applied Science, California Institute of Technology, Pasadena, CA, USA,
[*] School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA.

Keywords: Information Gathering; Active Learning; Submodular Functions; Observation Selection

1 Introduction

High levels of pollutants, such as nitrates, in lakes and rivers can lead to the rapid growth of algae. These algal blooms can be a severe threat to our drinking water resources. For example, the algal bloom at Taihu Lake, Jiangsu province, China in June 2007 deprived four million people from drinking water and required an estimated USD 14.5 billion of cleaning costs [27]. Many of the growth processes associated with such algal blooms are still not sufficiently well understood, and need to be studied in the lakes instead of the lab. A natural approach to obtain this understanding is to monitor environmental quantities such are temperature, nutrient distribution and fluorescence that are associated with such algal blooms. A promising technology for this goal is to use sensors carried by robotic boats to obtain measurements over a large spatial extent. However, since the time required to make each measurement is large, and the robot needs a certain amount of time to travel between sensing locations, it is an important to plan trajectories that allow us to obtain the most useful information about the phenomenon under study [33].

A similar problem arises in the context of monitoring municipal drinking water distribution networks. Accidental or malicious contaminations of such networks can affect a large population and cause significant harm. Such contaminations can potentially be detected by deploying sensors in the water distribution network. However, these sensors are fairly expensive (for example, the Hach Water Distribution Monitoring system may be a suitable solution, but is currently priced above USD 13,000 per unit). We must thus optimize the locations where these costly sensors are placed in order to maximize their effectiveness. The need for addressing this task has received significant attention, and recently the *Battle of the Water Sensor Networks* [29] was instituted as a benchmark challenge to fertilize research in this area.

More generally, in sensing optimization, our goal is to learn something about the state of the world (such as temperature at a given location, or whether there is a contamination), by optimally making a small set of expensive measurements. The fundamental question is: *How can we get the most useful information, at minimum cost?* This fundamental problem has been studied in several communities, including statistics (experimental design) [25, 14, 4, 30], machine learning (active learning) [26, 5, 3], operations research (facility location) [28, 1], sensor networks [39, 13, 12] and robotics [32, 36, 15]. However, most of the existing approaches rely on heuristics without guarantees, which can potentially perform arbitrarily poorly. There are also algorithms that are designed to find the optimal solution, such as algorithms for solving Partially Observable Markov Decision Processes (POMDPs) or Mixed Integer Programs (MIPs). However, these techniques are often very difficult to scale to large problems.

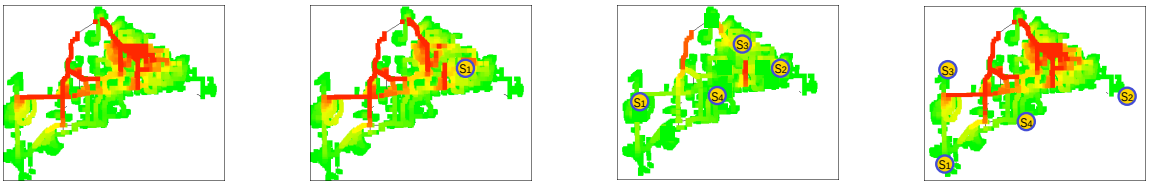
In this paper, we describe a new class of algorithms for addressing this fundamental question, which both have strong theoretical performance guarantees, and scale to large real sensing problems. Our algorithms are based on the key insight that many sensing problems satisfy *submodularity*, an intuitive diminishing returns property: Adding an observation helps more if we have made few observations so far than if we already have made many observations. Whenever a problem satisfies this diminishing returns property we can develop very effective algorithms that are both guaranteed to perform well in theory and work very well in practice. In addition to identifying the most informative sensing locations, our algorithms can handle more challenging settings, where sensors need to be able to reliably communicate over lossy links, where mobile robots are used for collecting data or where solutions need to be robust against adversaries and sensor failures.

In addition to introducing algorithms for solving submodular sensing problems, we also present results applying our algorithms to several real-world sensing tasks, including environmental monitoring using robotic sensors, activity recognition using a built sensing chair, and a sensor placement challenge. However, submodularity and our algorithms are not restricted to physical sensing problems. To illustrate this versatility, we also present results on deciding which blogs to read on the web in order to stay on top of the most important stories.

2 Sensing Optimization Problems

The sensing optimization problem seeks to find a set of sensors that provides the best possible sensing quality at the minimum possible cost; let us start by defining the notions of sensing quality and cost. Intuitively, a sensing quality function $F(\mathcal{A})$ provides a score for selecting the set of locations $\mathcal{A}$ out of the possible locations $\mathcal{V}$.

Consider the problem of protecting our water distribution systems from contaminations. Here, $F(\mathcal{A})$ measures, for example, the number of people protected by placing sensors at locations $\mathcal{A}$. To understand such a function better, consider the effect of introducing a contamination at some location i; as this contaminant spreads through the network, thousands or millions of people may be affected. Due to the complex dynamics of the system, contaminations at



(a) *Impact of contaminations* (b) *After one sensor placed* (c) *Effective placement* (d) *Poor placement*

Figure 1: Impact of contamination events on a large drinking water distribution network. The color of each node indicates the impact severity before detection if a contaminant is introduced at that node, and sensors are placed at the indicated locations. Red indicates high impact, green indicates low impact. In (a), no sensors are placed, and hence contamination is never detected.

some locations spread further than at others, as illustrated in Figure 1(a). Once we place a sensor, some of these contaminations are detected earlier (Figure 1(b)), and a number of people are protected from the contaminant. A good set of sensor placements $\mathcal{A}$, e.g., Figure 1(c), may detect many contaminations early, so $F(\mathcal{A})$ will be high. Conversely, a poor placement $\mathcal{A}'$, e.g., Figure 1(d), will have low $F(\mathcal{A}')$. In Section 4.1, we provide a more formal definition of $F(\mathcal{A})$ for this application.

In general, the goal of sensing optimization is to find a set $\mathcal{A}^*$ which provides the maximum possible value: $\mathcal{A}^* = \operatorname{argmax}_{\mathcal{A}} F(\mathcal{A})$ (or at least find a solution which is close to this optimal value). Intuitively, we can maximize $F(\mathcal{A})$ by placing sensors at every possible location, but, in real applications, we will have resource constraints, e.g., sensors cost money and we have a limited budget. The simplest type of constraint is a *cardinality constraint*: we have k sensors, and we want to ensure that $|\mathcal{A}| \leq k$. In this case, our optimization problem becomes:

$$\mathcal{A}^* = \operatorname*{argmax}_{\mathcal{A} \subset \mathcal{V}: |\mathcal{A}| \leq k} F(\mathcal{A}). \tag{1}$$

More generally, sensors may have different costs, e.g., drilling into a pipe may be more expensive than placing a sensor in an existing junction. Here, we denote the cost of a sensor s by $C(s)$; the cost of a set of sensors $\mathcal{A}$ is the sum of their individual costs, $C(\mathcal{A}) = \sum_{s \in \mathcal{A}} C(s)$. If we have we may have maximum budget B to spend, then our constraint becomes $C(\mathcal{A}) \leq B$. In this paper, we provide very practical algorithms with theoretical guarantees for such sensing problems, and for much more general ones.

3 Submodularity

Even the simplest formulation of these sensing problems in Equation 1 is already NP-hard [20], and it is unlikely that there will be an efficient algorithm for solving this problem exactly. However, consider what perhaps is the simplest possible heuristic: the *greedy algorithm*. This procedure starts by picking the element s_1 that provides the most information: $s_1 = \operatorname{argmax}_s F(s)$. Then, we iteratively pick elements s_i that provide the most additional information: $s_i = \operatorname{argmax}_s F(\{s\} \cup \{s_1, \ldots, s_{i-1}\})$.

This heuristic seems naïve, because, for example, when we pick the second sensor, the choice of first sensor is fixed and cannot be revised. This simple heuristic, however, performs surprisingly well in practice in many real world applications (e.g., Figure 3 for water distribution systems). In fact, for many practical applications it is hard to find an algorithm that performs significantly better than the greedy approach for the optimization problem in Equation 1. This empirical observation leads to a very interesting theoretical question: "*why does the greedy algorithm perform so well on sensor placement problems?*"

Interestingly, this question can be answered by introducing a structural property called *submodularity* that is present in many practical sensing optimization problems. Intuitively, a problem is submodular if we observe *diminishing returns*. For example, if we have deployed 5 sensors, a 6^{th} one will provide much additional information, while after deploying 500 sensors, the 501^{st} will provide much less new information. Figure 3 illustrates this concept using the classic notion of set cover. Diminishing returns can be seen throughout our lives, e.g., when applied to business practices, the Pareto Principle says that "80% of your sales come from 20% of your clients." In water distribution systems, we naturally see diminishing returns: a few sensors may protect a large portion of the population, but to protect the last few people, we may need many more sensors.

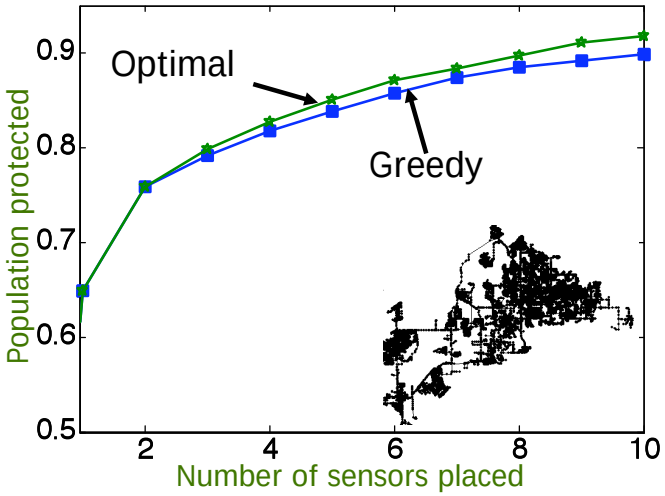
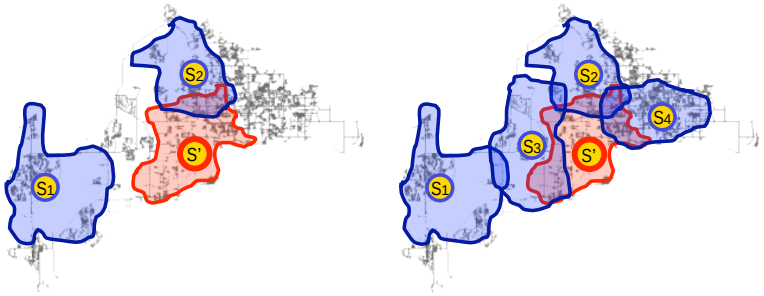


Figure 2: When placing sensors to maximize the expected detection performance, the greedy algorithm provides near-optimal performance.



(a) *Adding s' to a small deployment* (b) *Adding s' to a larger deployment*

Figure 3: Illustration of the diminishing returns effect (submodularity) of adding sensors. The blue regions indicate nodes where contamination is detected quickly using the existing sensors. The red region indicates the additional coverage by adding a new sensor s'. If more sensors are already placed (b), there is more overlap, hence less gain in sensing quality.

More formally, a set function is said to be submodular [28] if adding an element s to a set $\mathcal{A}$ provides more gain than adding it to a set $\mathcal{B}$, if $\mathcal{A}$ is a subset of $\mathcal{B}$, i.e., $\forall \mathcal{A} \subset \mathcal{B} \subset \mathcal{V}$ and $\forall s \in \mathcal{V} \setminus \mathcal{B}$, we have that $F(\{s\} \cup \mathcal{A}) - F(\mathcal{A}) \geq F(\{s\} \cup \mathcal{B}) - F(\mathcal{B})$. Many real world problems satisfy this property, e.g., the function measuring the population protected by sensing in our water distribution systems [20]. If a problem is submodular, we can exploit this property to develop very efficient algorithms that are guaranteed in theory to provide near-optimal solutions, and perform extremely well in practice.

To understand the power of this insight, let us consider the class of submodular functions that are monotonically increasing as the set size increases: $\forall \mathcal{A} \subset \mathcal{B} \subset \mathcal{V}$, we have that $F(\mathcal{A}) \leq F(\mathcal{B})$. Intuitively, population protected is monotonic: adding more sensors always protects more people, since some contaminations can be detected earlier. For monotonic submodular functions, the classic result of Nemhauser et al. [28] justifies why the greedy algorithm performs so well in practice:

Theorem 1 (Nemhauser et al. [28])**.** *For monotonic submodular functions, the quality of the set of elements $\mathcal{A}_{greedy}$ obtained by the greedy algorithm is within a constant factor of the optimal solution $\mathcal{A}^*$ to Equation 1:*

$$F(\mathcal{A}_{greedy}) \geq \left(1 - \frac{1}{e}\right) F(\mathcal{A}^*).$$

This result says that the quality of the solution provided by the greedy algorithm is at least 63% of the optimal quality ($1 - 1/e \approx 0.63$). In practice, the performance of the greedy algorithm is much closer to optimal, e.g., see Figure 3.

Furthermore, no other efficient algorithm can obtain better performance in general, unless P=NP [11]. These two facts provide a notable theoretical explanation as to why this simple heuristic performed so well in practice. In the remainder of this paper, we will provide a number of sensing optimization problems and applications. In many of these applications, the greedy algorithm is not near-optimal, but, by exploiting submodularity, we can design very efficient algorithms with near-optimal performance.

4 Case studies in physical sensing

We will now present two case studies, demonstrating real-world examples of submodular sensing problems.

4.1 Protecting municipal drinking water distribution networks

As introduced in our running example, accidental and malicious contamination in municipal drinking water distribution networks can pose a severe threat to the population. Such contamination could potentially be detected by deploying sensors in the junctions and pipes of the network. Existing sensor solutions are fairly expensive (sensors such as the Hach Water Distribution Monitoring system are currently priced above USD 13,000 per unit), and thus only small numbers of these sensors can be deployed. Hence optimizing their placement becomes of crucial importance. The problem of deploying a small number of sensors to optimize performance criteria such as time to detection or minimizing the expected population affected by contamination has received significant attention. During the 8th Annual Water Distribution Systems Analysis Symposium, Cincinnati, Ohio, in August 2006, a challenge (the *Battle of the Water Sensor Networks*, BWSN) was organized, in which the 13 participating teams competed for optimal performance [29]. The statement of this challenge included a realistic model of a real metropolitan area water distribution network (Figure 4(a)) with 12,527 nodes, as well as a description of 3.6 million realistic contamination scenarios, which varied in the choice of injection location and time of the contaminant into the network, as well as other parameters. The EPANET 2.0 simulator developed by the Environmental Protection Agency (EPA) [31] was used to estimate the impact of possible contaminations. The goal of the challenge was to optimize sensor deployments with respect to *multiple objectives*: Minimizing the time to detection, the expected population affected by contamination, the total amount of contaminated water consumed, and maximizing the detection likelihood. All of these objectives $F(\mathcal{A})$ can be written as

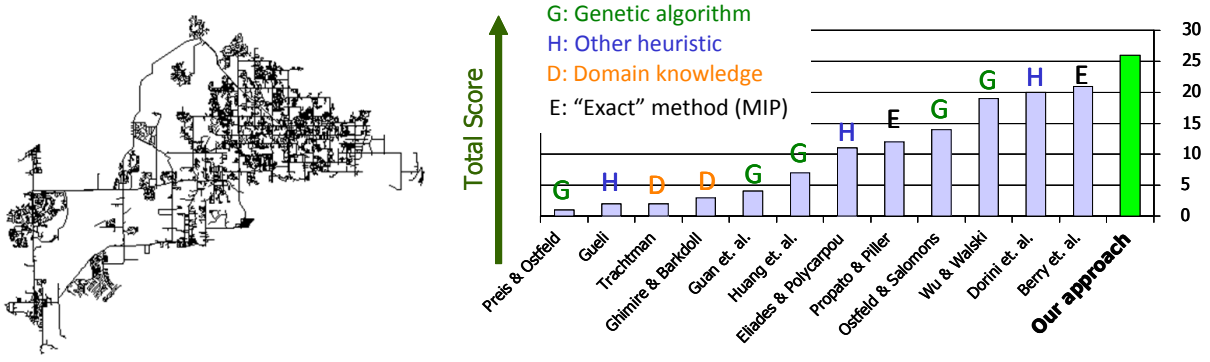
$$F(\mathcal{A}) = \sum_{i \in \mathcal{I}} p(i) \max_{s \in \mathcal{A}} M_{is}, \tag{2}$$

where $p(i)$ is the probability that contamination scenario $i \in \mathcal{I}$ occurs, and $M_{is} \geq 0$ is the amount of protection (for example, reduction in detection time, population affected, etc.) that placing a sensor $s \in \mathcal{V}$ provides in case contamination scenario $i \in \mathcal{I}$ occurs. As we show in [20], objective functions of these form and the ones used in the challenge are submodular. Our approach was to exploit this submodularity property to provide near-optimal sensor placements. However, evaluating the objective function $F(\mathcal{A})$ requires the values M_{is}, and thus the estimation of the impact of all 3.6 million contamination events $i \in \mathcal{I}$, along with the benefit of placing each sensor at each possible location. Due to the magnitude of the problem instance considered in the challenge (the largest water distribution network studied by the community by that time), this task required *massive amounts of distributed computation* (processing approximately 47 Terabyte of simulation data in a cluster of 20 multi-core machines [20]). In addition, evaluating each function $F(\mathcal{A})$ is thus very computationally expensive, and running the greedy algorithm to select 20 sensors takes approximately 30 hours in a highly optimized implementation. By exploiting submodularity, we were able to drastically reduce this computation time to approximately 1 hour using a technique called *lazy evaluations*, as shown in Figure 4.1 [20]. We call our algorithm that implements these lazy evaluations the CELF algorithm.

In order to evaluate the entries to the challenge, different parameter settings were varied in the problem instances, including the amount of contaminant introduced and the delay before detection. Altogether, contributions were evaluated in 30 different settings. Since the goal was to solve a multicriterion optimization problem, for each of these settings, the set of non-dominated entries was identified[1]. Each participating research team was evaluated based on the total number of non-dominated solutions (with 30 being the maximum possible score).

The participants in the challenge used different algorithms for optimization, including genetic algorithms and other heuristics, domain knowledge and exact solvers (such as approaches based on Mixed Integer Programming), that could be only applied to smaller parts of the problem instance due to its size. According to the performance evaluation by the

[1] hereby, an entry is dominated, if there is another entry that performs at least as well in all objectives, and strictly better in one of the objectives.



(a) *Water distribution network* (b) *BWSN Results*

Figure 4: (a) Illustration of the municipal water distribution network considered in the Battle of the Water Sensor Networks challenge. (b) Result of BWSN challenge.

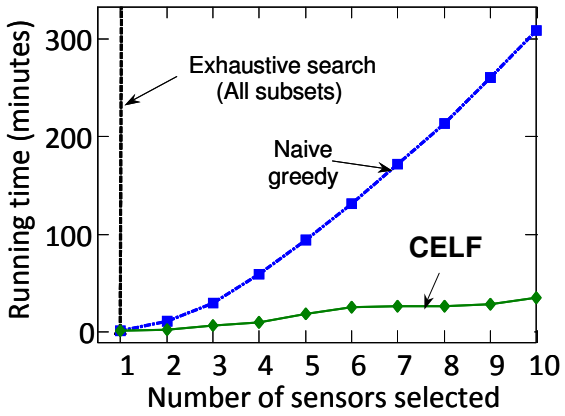


Figure 5: Running time comparison of different algorithms for sensor placement. The CELF algorithm provides a factor 30 speedup over the naive greedy algorithm.

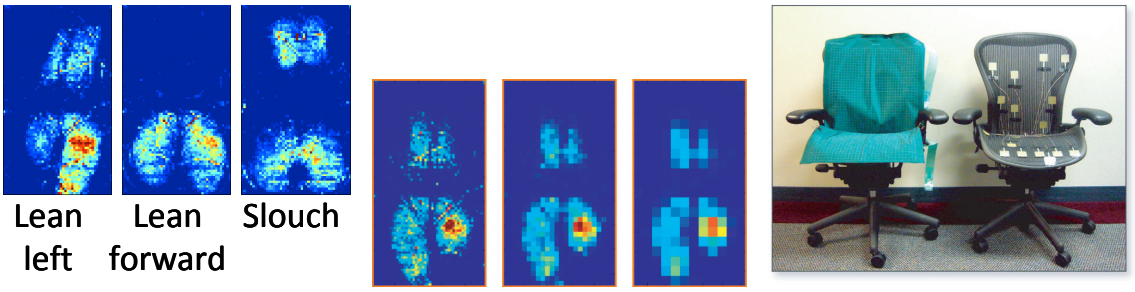
organizers of the challenge Figure 4(b) our solution based on submodular function optimization obtained 24% higher score than the runner-up [29].

4.2 Recognizing seating postures using a sensing chair

Many people spend a large part of their day sitting, be it in an office chair, couch or car seat. In these contexts, sustained poor sitting posture can cause significant adverse effects. For example, back pain affects 60% to 80% of adults in the U.S. at some time in time in their lives, causing expensive absence from work [9]. Elderly nursing home residents, who often spend long periods of time seated, are likely to develop ulcers [34]. Improved seating comfort for these individuals augmented with technology interventions may have significant impact on health productivity and injury prevention.

Posture is a primary measure of sitting comfort (and discomfort) [8]. Posture information could also help predict a sitter's state. For instance, subtle changes in seating posture (e.g., when talking to other people in the car) can indicate attentiveness; unattentive driving can lead to an increased chance of accidents. Hence, posture information can potentially be used to reduce severe adverse effects (health problems, decreased productivity or car accidents).

Previous work [37, 40] used a high-fidelity pressure sensor mat placed on the seat to acquire posture information. This sensor mat obtains pressure values on a 84×48 grid, as shown in Figure 6(c). Using this high-resolution sensor, previous work obtained 80% accuracy for classifying ten different postures (Figure 6(a)), such as leaning forward, leaning backward, slouching, etc. While these approaches demonstrated significant potential for using posture recognition as input modality, the sensor solution is very expensive, currently priced at above USD 3,000, precluding



(a) *Examples of postures* (b) *Choosing sensor types* (c) *Sensing chairs*

Figure 6: (a) Examples of pressure maps obtained if users assume different postures on the sensing chair. (b) Pressure maps under various types of pressure sensors varying in resolution. (c) The sensing chairs used in our experiments. Left: high resolution sensor mat. Right: optimized sensor placement. Note that sensors are not placed symmetrically on the chair; one possible reason is that, due to symmetries in the body, there is a diminishing returns effect for placing sensors at symmetric locations.

a widespread deployment of such sensing technology.

One approach to reduce the cost is to replace the expensive, high-resolution sensor with a small number of well placed sensors. In order to optimize the sensor deployment, we learn a probabilistic model $P(Y, \mathcal{X}_\mathcal{V})$ from training data collected by [37] from 20 users. This model encodes the statistical dependencies between the assumed posture Y and the pressure values $\mathcal{X}_s$ at all 4032 locations $s \in \mathcal{V}$ of the sensor mat. Based on this probabilistic model, we want to select the sensor locations $\mathcal{A}$ that maximize the *information gain* about the posture (i.e., maximally reduces the entropy of the posture predictor). Recently, we have shown that this information gain criterion $F(\mathcal{A}) = H(Y) - H(Y \mid \mathcal{X}_\mathcal{A})$ is monotonic and submodular for a large class of probabilistic models arising in many real-world applications [22]. Hence, we can use the greedy algorithm for selecting the sensor locations.

However, we do not only have the choice of *where* to place sensors, but also *which type* of sensors to deploy. More specifically, we face the decision of whether to deploy small, point sensors, or larger, flat sensors that average pressure over an extended area (*c.f.,* Figure 6(b)). In addition, each of these sensor types has a different cost, and our goal is to select a most cost-effective sensor solution. While the regular greedy algorithm introduced in Section 3 does not apply in the case of non-uniform cost functions, we can use the CELF algorithm [24], a variant of the greedy algorithm that we developed for cost-benefit optimization, to solve this more general optimization problem.

For various choices of the budget, we use the CELF algorithm to optimize the sensor placement, and study the estimated increase in classification accuracy. We choose a budget at a point where placing additional sensors does not significantly increase the the (cross-validated) classification accuracy, and thus arrive at a final design of 19 sensors.

In order to evaluate the performance of the proposed sensor design, we instrumented a new chair (Figure 6(c)). The cost of this proposed sensor design is approximately USD 100. We then perform a user study with 20 naive subjects. In this experiment, the optimized sensor design achieved 78%, as compared to 80% using the expensive sensor mat.

5 Robust sensing optimization

In Section 4.1, we considered the problem of selecting sensor locations in a water distribution network which effectively detect random, accidental contamination events (i.e., each scenario $i \in \mathcal{I}$ has a certain probability $p(i)$ of occurrence). However, if an adversary learns about the sensor locations, they can act strategically and maliciously contaminate the network based on that knowledge. In order to protect against such an adversary, it is important to decide where to place sensors in order to maximize the worst-case detection performance.

As we show in Section 5.3, this and many other problems can be formulated as optimizing a sensor deployment against an adversary who can, after seeing our deployment, select the minimum over a set of submodular functions. More formally, we pick $\mathcal{A}^*$ to maximize

$$\mathcal{A}^* = \operatorname*{argmax}_{\mathcal{A} \subset \mathcal{V}: |\mathcal{A}| \leq k} \min_i F_i(\mathcal{A}), \tag{3}$$

where $F_1, \ldots, F_m$ is a collection of monotonic submodular functions. In the water networks example, F_i is the detection performance that sensors $\mathcal{A}$ achieve if the adversary chooses contamination scenario $i \in \mathcal{I}$. In the following, we will describe an algorithm for solving such *robust* sensing problems.

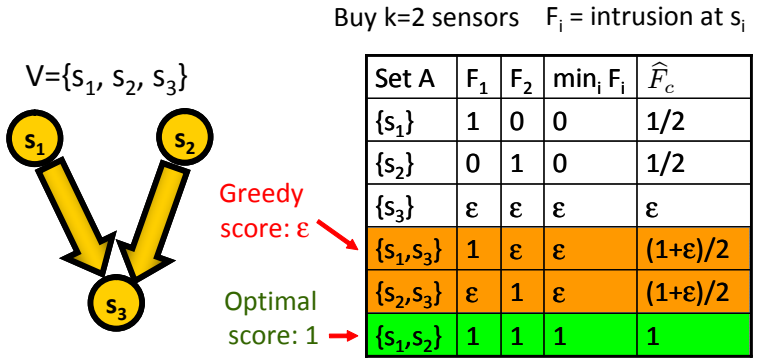


Figure 7: Example demonstrating poor performance of greedy algorithm for robust sensor placement. Left: Simple water distribution network with 3 nodes, where water flows downstream from s_1 and s_2 to s_3. Contaminations can occur at nodes s_1 and s_2. Node s_3 detects both events, but only after a long time. Node s_1 immediately detects contamination at s_1, but never at s_2, and vice versa. Thus, greedy algorithm picks s_3 first, as shown in the table on the right. The SATURATE algorithm (Section 5.2) optimizes the average truncated sensing quality $\widehat{F}_c(\mathcal{A})$. For $c = 1$, SATURATE selects s_1 first, and subsequently recovers the optimal solution.

5.1 When greedy algorithms fail

Given the provably near-optimal performance of the greedy algorithm for optimizing submodular functions, i.e., for solving Problem (1), a natural approach to the robust optimization problem (3) would be to modify the greedy algorithm, so that, after elements $s_1, \ldots, s_{j-1}$ have been selected, it adds the element

$$s_j = \operatorname*{argmax}_{s \in \mathcal{V}} \min_i F_i(\{s_1, \ldots, s_{j-1}\} \cup \{s\}),$$

e.g., in the water network example, it adds the sensor that most decreases the worst-case detection time.

Unfortunately, for the robust optimization problem (3), this greedy algorithm can fail arbitrarily badly. To see this, consider the following example, illustrated in Figure 5.1. Suppose there is two contamination events, i_1, i_2, and three possible sensor locations s_1, s_2 and s_3. A sensor at location s_1 can immediately detect event i_1, i.e., $F_1(\{s_1\}) = 1$, but never detect i_2, i.e., $F_1(\{s_2\}) = 0$. Similarly, $F_2(\{s_1\}) = 0$ and $F_2(\{s_2\}) = 1$. A sensor at location s_3 can detect both contamination events, but only after a long time, i.e., $F_1(\{s_3\}) = F_2(\{s_3\}) = \varepsilon$. Now suppose we can afford to place two sensors, i.e., $k = 2$. If we run the greedy algorithm, it will place a sensor at location s_3 first, since that is the only location s such that $\min_i F_i(\{s\}) > 0$. But once the greedy algorithm commits to picking s_3, it can only select s_1 or s_2, but not both, leaving one of the contamination events essentially unprotected. Thus, the greedy algorithm obtains a worst-case score of ε. On the other hand, the optimal solution is to place sensors at locations s_1 and s_2, obtaining a worst-case score of 1, which is arbitrarily better than ε.

One might think that this poor performance is a particular property of the greedy algorithm, and other algorithms might do better. Unfortunately, as we prove in [21], unless $P = NP$, there cannot exist any efficient algorithm that finds a set $\mathcal{A}$ of size k achieving *any* nontrivial approximation guarantee!

5.2 SATURATE Algorithm for robust sensing

The above is a very strong negative result – as long as one insists on picking at most k sensor locations, one cannot hope to efficiently recoup not even an exponentially small fraction of the optimal value achievable using k sensors. This insight motivates the question: *Can we achieve non-trivial guarantees if we allow ourselves to pick slightly more than k sensors?*

To address this problem, we have developed the SATURATE algorithm [21], which is guaranteed to efficiently find a solution $\mathcal{A}_S$ such that $\min_i F_i(\mathcal{A}_S) \geq OPT_k$, where OPT_k is the optimal score achievable using k sensors. Rather than using k sensors, however, SATURATE will use $|\mathcal{A}_S| \leq \alpha k$ for some small value $\alpha > 1$. Hence, SATURATE is guaranteed to find a solution which performs at least as well as the best solution of size k, but the returned solution is slightly larger than k.

The key idea of the SATURATE algorithm is the following: Suppose we can find an algorithm that, given an input value c either finds a solution $\mathcal{A}$ of size at most αk with $\min_i F_i(\mathcal{A}) \geq c$, or guarantees that no such solution exists.

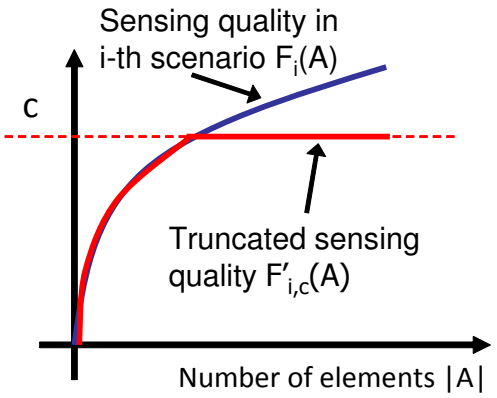


Figure 8: Truncation of sensing quality using in SATURATE algorithm.

We call such an algorithm an *approximate feasibility checker*. Such an algorithm would be extremely useful: if we give it any guess c on the optimal value OPT_k, it can either tell us that $c > OPT_k$, i.e., c is a strict upper bound to the optimal value, or it finds a solution that achieves this value c and is approximately feasible (i.e., of size at most αk). We can thus search for the optimal value c that is still approximately feasible. This search procedure can be efficiently implemented using a binary search strategy.

Hence, the main problem is to devise an approximate feasibility checker. One approach to develop such an algorithm is to attempt to solve the optimization problem

$$\min_{\mathcal{A}\subseteq\mathcal{V}} |\mathcal{A}| \text{ s.t. } \min_i F_i(\mathcal{A}) \geq c, \tag{4}$$

i.e., find the smallest possible set of sensors that achieves a worst-case performance of at least c. Unfortunately, this optimization has a non-submodular constraint in it, as it requires that $\min_i F_i(\mathcal{A}) \geq c$. Our key idea is to replace this non-submodular constraint by a submodular constraint. This can be done in the following way: For each function F_i, define its *truncation at level* c by $F_{i,c}(\mathcal{A}) = \min\{F_i(\mathcal{A}), c\}$, as illustrated in Figure 5.2. Interestingly, this truncation preserves monotonic submodularity. Now consider the *average truncated score*, $\widehat{F}_c(\mathcal{A}) = \frac{1}{m}\sum_i F_{i,c}(\mathcal{A})$. Since nonnegative linear combinations of submodular functions are still submodular, $\widehat{F}_c(\mathcal{A})$ is a submodular function. Now, note that

$$\min_i F_i(\mathcal{A}) \geq c \Leftrightarrow \widehat{F}_c(\mathcal{A}),$$

i.e., the worst-case score is at least c if and only if the average truncated score is at least c. Thus, instead of solving the non-submodular problem in Eq. (4), we can solve the equivalent submodular problem

$$\min_{\mathcal{A}\subseteq\mathcal{V}} |\mathcal{A}| \text{ s.t. } \widehat{F}_c(\mathcal{A}) \geq c. \tag{5}$$

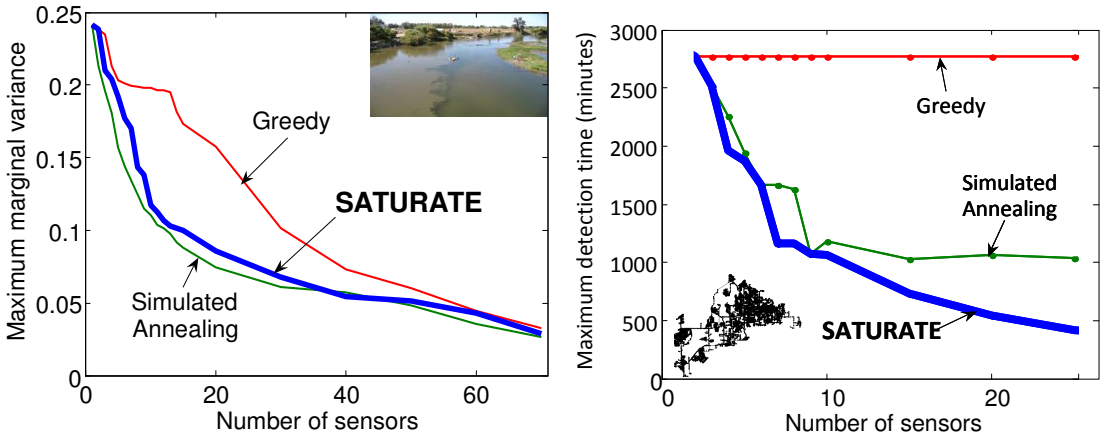
Surprisingly, this problem can be approximately solved using a variant of the greedy algorithm, which, instead of stopping after k sensors have been selected, stops after the score achieves value c. This greedy algorithm returns a set $\mathcal{A}_G$ such that $|\mathcal{A}_G| \leq \alpha|\mathcal{A}^*|$, where $\mathcal{A}^*$ is an optimal solution to Problem 5, and α depends logarithmically on the size of the problem instance.

This insight suggests a very simple approximate feasibility checker: Run the greedy algorithm on $\widehat{F}$, and check whether the resulting solution is of size greater than αk (in which case the optimal solution must have value less than c), or is at most αk (in which case an approximately feasible solution was found). When combined with the binary search strategy illustrated above, this procedure defines our SATURATE algorithm, which provides a near-optimal solution to the robust sensing problem:

Theorem 2. *Let $OPT_k = \max_{|\mathcal{A}|\leq k} \min_i F_i(\mathcal{A})$ be the optimal robust value for placing k sensors. Then* SATURATE *finds a solution $\mathcal{A}_S$ that achieves at least this value:*

$$\min_i F_i(\mathcal{A}_S) \geq OPT_k,$$

using slightly more sensors, $|\mathcal{A}_S| \leq \alpha k$, where $\alpha = 1 + \log\left(\max_{s\in\mathcal{V}} \sum_i F_i(\{s\})\right)$.



(a) *Environmental monitoring*

(b) *Water distribution network*

Figure 9: Performance of SATURATE for environmental monitoring (a) and water networks (b). SATURATE drastically outperforms the greedy algorithm, and is competitive with (a) / superior to (b) highly fine-tuned simulated annealing algorithm.

In the example of Figure 5.1, when eventually considering the truncation $c = 1$, SATURATE in fact recovers the optimal solution.

One may ask whether one can improve on the logarithmic dependence of α on the size of the problem instance. As we have shown recently [21], this dependence is necessary under reasonable complexity-theoretic assumptions. Hence, in a strong sense, the SATURATE algorithm is a best-possible efficient algorithm for robust submodular optimization.

5.3 Cases studies on robust sensing

We have evaluated the SATURATE algorithm on several real world sensing problems. The first application is in environmental monitoring. Problems such as monitoring algal blooms in lakes, as introduced in Section 1, require estimating spatial phenomena such as temperature, fluorescence and nutrient distribution over a large spatial extent. One approach to estimate phenomena such as temperature is to make measurements at a small number of locations (e.g., using sensor buoys or sensors carried by robotic boats), and then predict the temperature at the unobserved locations using statistical models. Gaussian Processes (also known as Kriging models) have been found very effective for that purpose [6]. However, in order to obtain accurate predictions, we must select measurement locations that minimize the expected error in the predictions made given these measurements.

One approach to quantify the prediction error at some location $s \in \mathcal{V}$ is the expected posterior variance $\mathrm{Var}(\mathcal{X}_s \mid \mathcal{X}_\mathcal{A})$ given that we have made measurements at locations $\mathcal{A}$. In many cases, the reduction in variance at location s, $F_s(\mathcal{A}) = \mathrm{Var}(\mathcal{X}_s) - \mathrm{Var}(\mathcal{X}_s \mid \mathcal{X}_\mathcal{A})$ can be shown to be a submodular function [7]. Optimizing the average prediction error throughout the environment (e.g., lake) is a natural (submodular) function we can optimize. Unfortunately, such an average-case optimization does not guarantee good predictions at every point in the lake, which could lead us to miss an important event. A good alternative is to optimize the worst-case prediction error over the entire lake; a problem often called *minimax kriging* [2]. Thus, selecting a set of locations to measure in order to minimize the worst-case prediction error is an example of a robust submodular sensing problem. We use SATURATE algorithm on this problem, and compare it against the state of the art from geostatistics, a highly fine-tuned simulated annealing algorithm, which has seven parameters that need to be carefully fine-tuned for the particular application at hand [38]. Figure 9(a) compares the performance of the algorithms, on a problem of predicting pH values in a lake near Merced, California. We can see that the greedy algorithm performs very poorly on this task. The simulated annealing algorithm performs much better, requiring approximately half as many samples to achieve the same worst-case error. The SATURATE algorithm is competitive with the state of the art, while being much simpler to implement, requiring no parameters to tune and running approximately 10 times faster in our experiment.

In some sense, the worst-case prediction problem for the river data is not very challenging, since the greedy algorithm does not perform arbitrarily badly, as the theory would predict. We get a very different outcome when we compared the performance of the algorithms on the problem of deploying sensors in drinking water distribution networks in order to minimize the worst-case time to detection. Figure 9(b) shows the results of this experiment.

Figure 10: The aquatic Networked Infomechanical System (NIMS-AQ) [16] for environmental monitoring.

Interestingly, in this domain, the greedy algorithm does not manage to decrease the worst-case detection time at all. The explanation of this poor performance is that unless all contamination events are detected, the worst-case detection time remains the same. However, no single sensor can detect all contamination events, thus the greedy algorithm has no indication of which first sensor location to pick. The simulated annealing algorithm randomizes, so eventually it gets lucky and obtains non-trivial performance. However, simulated annealing performs drastically worse than the SATURATE algorithm, which obtains 60% lower worst-case detection time when placing 25 sensors.

6 Tackling constraints of real applications

So far, we have seen that many real-world sensing problems require maximization of a submodular function. However, each application presents its own challenges. For example, when deploying networks of wireless sensors, the chosen sensor locations cannot be too far apart, otherwise the sensors will not be able to communicate through wireless links. Similarly, if we use robotic sensors to make observations, the chosen sensor locations need to be connected by a path of bounded length, which is then traversed by the robot with limited battery power. These two problems can be modeled by using complex cost functions: We can consider all sensor locations as nodes in a graph, and the cost $C(\mathcal{A})$ of any set of locations $\mathcal{A}$ as the cost of the cheapest routing tree (when deploying wireless networks) or the cheapest path (when using robotic sensors) that connects the chosen locations in the graph. In these settings, each initial decision has a large effect on the cost of subsequent observations. For this reason, the greedy algorithm, which performs near-optimally for the problem of selecting the best k locations (i.e., $C(\mathcal{A}) = |\mathcal{A}|$), can perform arbitrarily badly in these more complex settings. To address this challenge, we developed the PSPIEL and EMIP algorithms, which provide solutions that perform near-optimally both in terms of utility $F(\mathcal{A})$ and cost $C(\mathcal{A})$ [19, 33]. In a sensor placement study, our PSPIEL algorithm for *padded Sensor Placements at Informative and Cost-effective Locations* found a sensor placement of measuring light in buildings which outperformed a manual (more or less uniformly-placed) sensor network deployment, leading to 44% lower prediction error and 19% lower communication cost. We also used our EMIP algorithm for *efficient Multi-robot Informative Path planning* to plan informative paths for the NIMS-AQ robot [16] (Figure 6) in order to monitor electrical conductivity and temperature in the San Joaquin river and in a lake near the University of California, Merced. In these experiments, our informative path planning approach led to a drastic reduction in experimentation time for minimal loss in prediction accuracy.

Another fundamental constraint associated with wireless sensor networks is limited battery power: Every measurement and every message sent over the network drains the battery, leading to reduced network lifetime. One possible approach to increasing the lifetime of a deployed sensor network is *sensor scheduling*, activating sensors only infrequently as needed. Hence, in order to optimize sensor network performance, one has to decide both *where* to place the sensors and *when* to activate them. The traditional approach to this problem is to first decide where to place the sensors, and then optimize the sensor schedule. However, when battery constraints require sensor scheduling, a possibly more effective approach would be to *simultaneously* optimize both the placement and the schedule. We recently developed the ESPASS algorithm for *efficient Simultaneous Placement and Scheduling of Sensors*, which provides near-optimal performance for this simultaneous optimization problem [23]. We evaluate our algorithm on several monitoring problems, and demonstrate significant advantages over the traditional, stagewise approach, such as a 30% improvement in

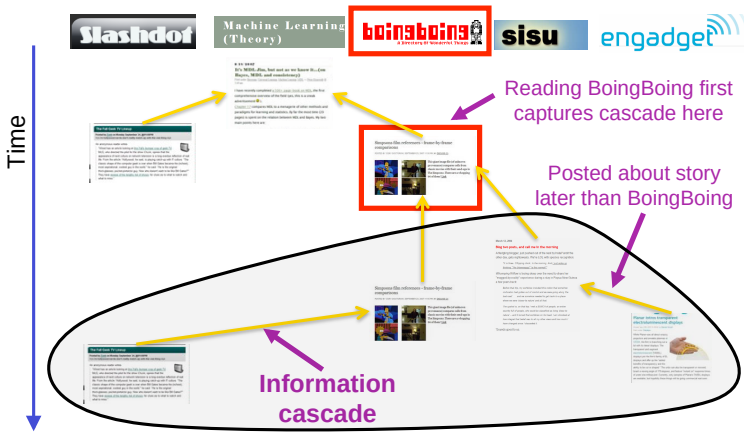


Figure 11: Illustration of an information cascade.

sensor network lifetime for the same prediction accuracy in a traffic monitoring application.

All the algorithms introduced above assume that a model of the world is available that allows us to predict the sensing quality obtained by making any set $\mathcal{A}$ of measurements. However, in many applications, such a model may not be available a priori. In such applications, we need to obtain measurements both to estimate the model, and to make accurate predictions using the estimated model. This problem shares similarities with the *exploration–exploitation* dilemma in reinforcement learning, where the goal is to choose actions to explore (refine the model of the world) and exploit (maximize the reward based on the estimated model). Analogously, we have developed an algorithm for addressing this exploration–exploitation tradeoff in environmental monitoring [18]. Our EEGP algorithm (for *Exploration–Exploitation in Gaussian Processes*) provides logarithmic sample complexity for estimating the parameters of the underlying Gaussian Process. We demonstrate the algorithm on several environmental monitoring problems, and show that it leads to drastic reduction in prediction error when compared to passive (exploitation-only) approaches (e.g., a 65% reduction in RMS when monitoring pH in Merced river).

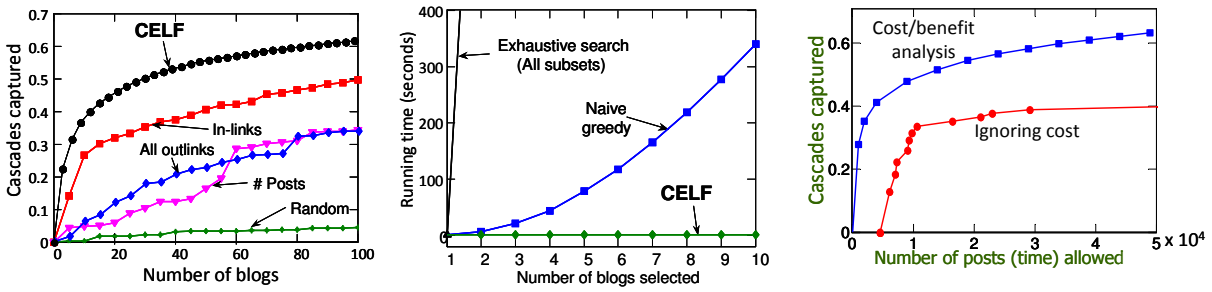
7 From water to the web: What blogs should I read?

Thus far, the case studies we have used have focused on optimization problems related to physical sensing; we will now address a very different problem: *sensing on the web*. In 2008, TIME Magazine asked: "How many blogs does the world need?" [17], claiming that there were already too many out there. The blogosphere has grown at a tremendous rate, with tens of millions of active blogs generating hundreds of millions of postings per year [35]. This huge activity leads to huge overload of information, opening up significant sensing optimization questions like: *if you only have 10 minutes a day to spend on the blogosphere, which blogs should you read in order to keep track of the big stories?*

To understand how to address this question, consider how information spreads through the blogosphere. As illustrated in Figure fig:cascade, a story posted in some blog may be picked up (and linked to) by other blogs; from there, these postings may be linked to by yet more blogs, and so on. Such spread of information in the blogosphere forms what we call an *information cascade* [24]. A good blog is one that captures big stories (i.e., large cascades) early on (i.e., as close to the first source as possible).

At first, the problem of capturing information cascades seems quite different from the other tasks we have discussed thus far. In reality, however, the spread of contaminants through water distribution systems is very similar to the spread of information through the blogosphere. And, most importantly, both of these tasks can be formulated as submodular optimization problems, which can be tackled by the algorithms we have described thus far.

As an example, let us formalize one criterion for characterizing how well a blog captures a cascade i sparked by a certain story: Consider reading a particular blog b, which discussed this story at a certain time t. If blog b captures the story early on in cascade i, and the story becomes huge (i.e., the cascade is large), then there will be many other postings in cascade i appearing at a time later than t. In this case, blog b was very successful at capturing cascade i. On the other hand, if you have a story which does not become popular, its cascade i' will be small. If blog b captures



(a) *Comparison to heuristics* (b) *Running time on blogs* (c) *Number of posts cost for blogs*

Figure 12: Performance of CELF algorithm on blog selection problem. (a) CELF outperforms heuristics. (b) Lazy evaluations give factor 700 speedup. (c) Cost-benefit optimization.

cascade i' late, few blogs will report on this story after blog b, but you will not lose much by being late, since the story was small in the first place. More formally, if we recommend a set $\mathcal{A}$ of blogs, the resulting value $F(\mathcal{A})$ will be:

$$F(\mathcal{A}) = \sum_{i \in \mathcal{I}} \max_{b \in \mathcal{A}} M_{ib}, \tag{6}$$

where M_{ib} is the set of postings on cascade i that are dated later than the time blog b first appears in this cascade. Note the similarity between this objective and the one for water distribution systems in Equation 2; most interestingly, we note that both objectives are submodular functions.

To demonstrate this algorithm in practice, let us consider a blog dataset from 2006, with 45,000 blogs, 10.5 million posts, and 16.2 million links (30 GB of data) [24]. In this dataset, we discovered 17,589 significant cascades, where each blog participates in 9.4 different cascades on average. As Figure 12(a) shows, the CELF algorithm optimizing Equation 6 greatly outperforms a number of other heuristic selection techniques. The best runner up, performing about 45% worse than CELF, pick blogs by the number of in- or out-links from/to other blogs in the dataset, which is the type of heuristic used by many blog search websites. Figure fig:blogs-running-time plots the running time of selecting k blogs. We see that exhaustively enumerating all possible subsets of k elements is infeasible (the line jumps out of the plot for $k = 3$). The simple greedy algorithm scales as $\Omega(k|\mathcal{V}|)$, since for every increment of k we need to consider selecting all remaining $|\mathcal{V}| - k$ blogs. The performance of our CELF algorithm is significantly better, e.g., for selecting 100 blogs, greedy algorithm runs 4.5h, while CELF takes 23 seconds (700 times faster).

The results presented so far assume that every blog has the same cost. Under this *unit cost* model, the algorithm tends to pick large, influential blogs, that have many posts. For example, `instapundit.com` is the best blog, but it has 4,593 posts. Interestingly, most of the blogs among the top 10 are politics blogs: `instapundit.com`, `michellemalkin.com`, `blogometer.nationaljournal.com`, and `sciencepolitics.blogspot.com`. Some popular aggregators of interesting things and trends on the blogosphere are also selected: `boingboing.net`, `themodulator.org` and `bloggersblog.com`. The top 10 blogs had more than 21,000 thousand posts in 2006. Under the unit cost model, large blogs are important, but reading a blog with many posts is time consuming. This motivates the *number of posts (NP)* cost model, where we set the cost of a blog to the number of posts it had in 2006.

Comparing the NP cost model with the unit cost in Figure 12(c), we note that under the unit cost model, CELF chooses expensive blogs with many posts. For example, to obtain the same objective value, one needs to read 10,710 posts under unit cost model. The NP cost model achieves the same score while reading just 1,500 posts. Thus, optimizing the benefit cost ratio leads to drastically improved performance. Interestingly, the solutions obtained under the NP cost model are very different from the unit cost model. Rather than picking large blogs which capture many stories, under the NP model, the algorithm discovered the notion of summarizer blogs (e.g., `themodulator.org`, `watcherofweasels.com`, `anglican.tk`); these bloggers navigate the blogosphere and discover and talk about stories that eventually become big. Blogs selected under NP cost appear about 3 days later in the cascade as those selected under unit cost, which further suggests that that summarizer blogs tend to be chosen under NP model.

We can also reverse our information management goal: rather than selecting blogs, we can find a principled approach for picking a set of posts that best covers the important stories in the blogosphere. This task can also be formalized as a submodular optimization problem, which is amenable to the algorithms described herein [10]. As an

(a) *Unpersonalized* (b) *Sports fan*

Figure 13: Important words in blog postings on January 21, 2009, before (a) and after (b) personalization. This figures were generate using *wordle.net*.

example, here is the set of posts that our algorithm selects for an eight hour period on January 18, 2009, if our budget k is set to five:

1. Israel unilaterally halts fire as rockets persist
2. Downed jet lifted from ice-laden Hudson River
3. Israeli-trained Gaza doctor loses three daughters and niece to IDF tank shell
4. EU wary as Russia and Ukraine reach gas deal
5. Obama's first day as president: prayers, war council, economists, White House reception

The selected five posts all cover important stories from this particular day.

Since people have varied interests, the ideal coverage algorithm should incorporate user preferences in order to tailor the selected posts to individual tastes. For this task, we must *learn a personalized coverage function* for the blogosphere through user interaction, e.g., from the user labeling posts they would like to read or not read, or by what posts the user clicked on [10]. As an example, Figure 13(a) illustrates the important words being discussed in the blogosphere on January 21^{st}, 2009, while Figure 13(b) shows the important words for the articles the algorithm picks for a user who is very interested in sports. Such a personalization problem exemplifies problems where, rather than being given a submodular function as we have discussed thus far, we need to address the problem of *learning a submodular function*.

8 Conclusions

Our society is drowning in information: The internet allows us to access virtually every bit of information available to humanity, at any given time and at virtually no cost. The ubiquitous availability of sensors allows monitoring of the physical world in an unprecedented manner. This development presents exciting new opportunities for the advancement of science and society. However, this explosion in availability of information also creates the fundamental challenge of developing new techniques for extracting the most useful information. We believe that the algorithms described in this paper present an interesting step in this direction.

Acknowledgements

This work was partially supported by NSF Grants No. CNS-0509383, CNS-0625518, and ARO MURI W911NF0710287. Carlos Guestrin were partly supported by an Alfred P. Sloan Fellowship, by an IBM Faculty Fellowship and an ONR Young Investigator Award N00014-08-1-0752. Andreas Krause was partly supported by a Microsoft Research Graduate Fellowship.

References

[1] J. Berry, W. Hart, and et.al. A facility location approach to sensor placement optimization. In *8th Annual Symposium on Water Distribution Systems Analysis*, Cincinnati, Ohio, 2006.

[2] T.M. Burgess, R. Webster, and A.B. McBratney. Optimal interpolation and isarithmic mapping of soil properties. iv. sampling strategy. *Journal of Soil Science*, 32:643–659, 1981.

[3] R. Castro, R. Willett, and R. Nowak. Faster rates in regression via active learning. In *NIPS*, 2005.

[4] K. Chaloner and I. Verdinelli. Bayesian experimental design: A review. *Statistical Science*, 10(3):273–304, Aug. 1995.

[5] David A. Cohn, Zoubin Gharamani, and Michael I. Jordan. Active learning with statistical models. *J AI Research*, 4:129–145, 1996.

[6] N. A. C. Cressie. *Statistics for Spatial Data*. Wiley, 1991.

[7] A. Das and D. Kempe. Algorithms for subset selection in linear regression. In *Symposium on the Theory of Computing*, 2008.

[8] Michiel P. de Looze, Lottie F. M. Kuijt-Evers, and Jaap van Dieen. Sitting comfort and discomfort and the relationships with objective measures. *Ergonomics*, 46(10):985–997, August 2003.

[9] A. Lahad; A. D. Malter; A. O. Berg; R. A. Deyo. The effectiveness of four interventions for the prevention of low back pain. *JAMA*, 272(1286-1291), 1994.

[10] Khalid El-Arini, Gaurav Veda, Dafna Shahaf, and Carlos Guestrin. Turning down the noise in the blogosphere. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, June 2009.

[11] Uriel Feige. A threshold of ln n for approximating set cover. *Journal of the ACM*, 45(4):634 – 652, July 1998.

[12] H. H. Gonzalez-Banos and J. Latombe. A randomized art-gallery algorithm for sensor placement. In *Proc. 17th ACM Symposium on Computational Geometry*, pages 232–240, 2001.

[13] Himanshu Gupta, Samir R. Das, and Quinyi Gu. Connected sensor cover: Self-organization of sensor networks for efficient query execution. In *MobiHoc*, 2003.

[14] R. A. Howard. Information value theory. In *IEEE Transactions on Systems Science and Cybernetics (SSC-2)*, 1966.

[15] Shihao Ji, Ronald Parr, and Lawrence Carin. Non-myopic multi-aspect sensing with partially observable markov decision processes. *IEEE Transactions on Signal Processing*, 55(6):2720–2730, June 2007.

[16] Brett L. Jordan, Maxim A. Batalin, and William J. Kaiser. NIMS RD: A rapidly deployable cable based robot. In *IEEE International Conference on Robotics and Automation (ICRA)*, Rome, Italy, April 2007.

[17] Michael Kinsley. How many blogs does the world need? *TIME Magazine*, 172(22), December 2008.

[18] A. Krause and C. Guestrin. Nonmyopic active learning of gaussian processes: An exploration–exploitation approach. In *ICML*, 2007.

[19] A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg. Near-optimal sensor placements: Maximizing information while minimizing communication cost. In *Proceedings of the Fifth International Symposium on Information Processing in Sensor Networks (IPSN)*, 2006.

[20] A. Krause, J. Leskovec, C. Guestrin, J. VanBriesen, and C. Faloutsos. Efficient sensor placement optimization for securing large water distribution networks. *Journal of Water Resources Planning and Management*, 136(6), 2008.

[21] A. Krause, B. McMahan, C. Guestrin, and A. Gupta. Robust submodular observation selection. *Journal of Machine Learning Research*, 9:2761–2801, December 2008.

[22] Andreas Krause and Carlos Guestrin. Near-optimal value of information in graphical models. In *UAI*, 2005.

[23] Andreas Krause, Ram Rajagopal, Anupam Gupta, and Carlos Guestrin. Simultaneous placement and scheduling of sensors. In *Information Processing in Sensor Networks*, 2009.

[24] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. Cost-effective outbreak detection in networks. In *13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007.

[25] D. V. Lindley. On a measure of the information provided by an experiment. *Annals of Mathematical Statistics*, 27:986–1005, 1956.

[26] D. MacKay. Information-based objective functions for active data selection. *Neural Computation*, 4(4):590–604, 1992.

[27] MSNBC. China to spend $14.5 billion to clean up lake. `http://www.msnbc.msn.com/id/21498294/`, June 2007.

[28] G. Nemhauser, L. Wolsey, and M. Fisher. An analysis of the approximations for maximizing submodular set functions. *Mathematical Programming*, 14:265–294, 1978.

[29] Avi Ostfeld, James G. Uber, Elad Salomons, Jonathan W. Berry, William E. Hart, Cindy A. Phillips, Jean-Paul Watson, Gianluca Dorini, Philip Jonkergouw, Zoran Kapelan, Francesco di Pierro, Soon-Thiam Khu, Dragan Savic, Demetrios Eliades, Marios Polycarpou, Santosh R. Ghimire, Brian D. Barkdoll, Roberto Gueli, Jinhui J. Huang, Edward A. McBean, William James, Andreas Krause, Jure Leskovec, Shannon Isovitsch, Jianhua, Carlos Guestrin, Jeanne VanBriesen, Mitchell Small, Paul Fischbeck, Ami Preis, Marco Propato, Olivier Piller, Gary B. Trachtman, Zheng Yi Wu, and Tom Walski. The battle of the water sensor networks (bwsn): A design challenge for engineers and algorithms. *submitted to Journal of Water Resources Planning and Management*, 2008.

[30] H. Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58:527–535, 1952.

[31] L A Rossman. The epanet programmer's toolkit for analysis of water distribution systems. In *Annual Water Resources Planning and Management Conference*, 1999.

[32] R. Sim and N. Roy. Global a-optimal robot exploration in slam. In *ICRA*, 2005.

[33] Amarjeet Singh, Andreas Krause, Carlos Guestrin, and William Kaiser. Efficient informative sensing using multiple robots. *Journal of Artificial Intelligence Research*, 34:707–755, April 2009.

[34] D. M. Smith. Pressure ulcers in the nursing home, [review]. *Annals Int. Med.*, 123(6):43342, 1995.

[35] Spinn3r. Rss content, news feeds, news content, news crawler and web crawler apis. `http://spinn3r.com/`, 2007.

[36] C. Stachniss, G. Grisetti, and W. Burgard. Information gain-based exploration using rao-blackwellized particle filters. In *RSS*, 2005.

[37] Hong Z. Tan, Lynne A. Slivovsky, and Alex Pentland. A sensing chair using pressure distribution sensors. *IEEE/ASME Transactions on Mechatronics*, 6(3):261–268, 2001.

[38] D. P. Wiens. Robustness in spatial studies ii: minimax design. *Environmetrics*, 16:205–217, 2005.

[39] F. Zhao, J. Shin, and J. Reich. Information-driven dynamic sensor collaboration for tracking applications. *IEEE Signal Processing*, 19(2):61–72, 2002.

[40] Manli Zhu, Aleix M. Martinez, and Hong Z. Tan. Template-based recognition of static sitting postures. In *Workshop on Computer Vision and Pattern Recognition for Human Computer Interaction, CVPR*, 2003.

Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA15213

Carnegie Mellon